Project-4 Neural Networks

If this project is completed independently, the student will receive bonus points.

Write an AI to identify which traffic sign appears in a photograph.

```
$ python traffic.py gtsrb
Epoch 1/10
500/500 [==============================] - 5s 9ms/step - loss: 3.7139 - accura
Epoch 2/10
500/500 [==============================] - 6s 11ms/step - loss: 2.0086 - accur
Epoch 3/10
500/500 [==============================] - 6s 12ms/step - loss: 1.3055 - accur
Epoch 4/10
500/500 [==============================] - 5s 11ms/step - loss: 0.9181 - accur
Epoch 5/10
500/500 [==============================] - 7s 13ms/step - loss: 0.6560 - accur
```

Background

As research continues in the development of self-driving cars, one of the key challenges is computer vision, allowing these cars to develop an understanding of their environment from digital images. In particular, this involves the ability to recognize and distinguish road signs – stop signs, speed limit signs, yield signs, and more.

In this project, you'll use TensorFlow to build a neural network to classify road signs based on an image of those signs. To do so, you'll need a labeled dataset: a collection of images that have already been categorized by the road sign represented in them.

Several such data sets exist, but for this project, we'll use the German Traffic Sign Recognition Benchmark (GTSRB) dataset, which contains thousands of images of 43 different kinds of road signs.

Getting Started

- Download the distribution code `projects/project4.zip` from files and unzip it.

- Download the `projects/gtsrb.zip` from files for this project and unzip it. Move the resulting `gtsrb` directory inside of your `project4` directory.
- Inside of the `project4` directory, run `pip3 install -r requirements.txt` to install this project's dependencies: `opencv-python` for image processing, `scikit-learn` for ML-related functions, and `tensorflow` for neural networks.

Guideline

First, take a look at the data set by opening the `gtsrb` directory. You'll notice 43 subdirectories in this dataset, numbered `0` through `42`. Each numbered subdirectory represents a different category (a different type of road sign). Within each traffic sign's directory is a collection of images of that type of traffic sign.

Next, take a look at `traffic.py`. In the `main` function, we accept as command-line arguments a directory containing the data and (optionally) a filename to which to save the trained model. The data and corresponding labels are then loaded from the data directory (via the `load_data` function) and split into training and testing sets. After that, the `get_model` function is called to obtain a compiled neural network that is then fitted on the training data. The model is then evaluated on the testing data. Finally, if a model filename was provided, the trained model is saved to disk.

The `load_data` and `get_model` functions are left to you to implement.

Your Tasks

1. **Complete the implementation of `load_data` and `get_model` in `traffic.py`.**
2. **Write a report showing the algorithm introduction and how your model is built, as well as all the experimental details and results.** If this project is done by a team work, please indicate the contribution of each team member in detail.

Specification

- The `load_data` function should accept as an argument `data_dir`, representing the path to a directory where the data is stored, and return image arrays and labels for each image in the data set.

    - You may assume that `data_dir` will contain one directory named after each category, numbered `0` through `NUM_CATEGORIES - 1`. Inside each category directory will be some number of image files.
    - Use the OpenCV-Python module (`cv2`) to read each image as a `numpy.ndarray` (a `numpy` multidimensional array). To pass these images into a neural network, the images will need to be the same size, so be sure to resize each image to have width `IMG_WIDTH` and height `IMG_HEIGHT`.
    - The function should return a tuple `(images, labels)`. `images` should be a list of all of the images in the data set, where each image is represented as a `numpy.ndarray` of the appropriate size. `labels` should be a list of integers, representing the category number for each of the corresponding images in the `images` list.
    - Your function should be platform-independent: that is to say, it should work regardless of operating system. Note that on macOS, the `/` character is used to separate path components, while the `\` character is used on Windows. Use `os.sep` and `os.path.join` as needed instead of using your platform's specific separator character.
- The `get_model` function should return a compiled neural network model.
    - You may assume that the input to the neural network will be of the shape `(IMG_WIDTH, IMG_HEIGHT, 3)` (that is, an array representing an image of width `IMG_WIDTH`, height `IMG_HEIGHT`, and `3` values for each pixel for red, green, and blue).
    - The output layer of the neural network should have `NUM_CATEGORIES` units, one for each of the traffic sign categories.
    - The number of layers and the types of layers you include in between are up to you. You may wish to experiment with:
        - different numbers of convolutional and pooling layers
        - different numbers and sizes of filters for convolutional layers
        - different pool sizes for pooling layers
        - different numbers and sizes of hidden layers
        - dropout
- In a separate file called *README.md*, document (in at least a paragraph or two) your experimentation process. What did you try? What worked well? What didn't work well? What did you notice?

Ultimately, much of this project is about exploring documentation and investigating different options in `cv2` and `tensorflow` and seeing what results you get when you try

them!

You should not modify anything else in `traffic.py` other than the functions the specification calls for you to implement, though you may write additional functions and/or import other Python standard library modules. You may also import `numpy` or `pandas`, if familiar with them, but you should not use any other third-party Python modules. You may modify the global variables defined at the top of the file to test your program with other values.

Hints

- Check out the official Tensorflow Keras overview for some guidelines for the syntax of building neural network layers. You may find the lecture source code useful as well.
- **You are allowed to use Pytorch to complete this project.** However, you have to switch to the Pytorch environment and write the corresponding code by yourself.
- The OpenCV-Python documentation may prove helpful for reading images as arrays and then resizing them.
- Once you've resized an image `img`, you can verify its dimensions by printing the value of `img.shape`. If you've resized the image correctly, its shape should be `(30, 30, 3)` (assuming `IMG_WIDTH` and `IMG_HEIGHT` are both `30`).
- If you'd like to practice with a smaller data set, you can download a modified dataset that contains only 3 different types of road signs instead of 43.

Submission

The **pdf** version of your project report and the source code are required to be submitted to Canvas. Please package and zip all your files as
`project4-<NetID>.zip`
For example, if your NetID is "my390", please name your file as "project4-my390.zip". **For your report, please make sure the file format is pdf. Otherwise, the report will not be reviewed.** For the source code, please submit all the completed files.